САМОВЧИТЕЛЬ З VBScript

Практичний курс від нуля до системної автоматизації Windows

Файли • папки • COM • WSH • WMI • реєстр • мережа • логування • автоматизація • відладка

Важливо: VBScript є legacy-технологією. Microsoft позначає його як deprecated; у майбутніх версіях Windows він має перейти у Feature on Demand перед остаточним вилученням. Цей підручник корисний для підтримки існуючих .vbs/.wsf-рішень та розуміння старої Windows-автоматизації.

Українською мовою • 2026

Зміст. Як користуватися самовчителем

Частина I. Старт та фундамент

1. Що таке VBScript і де він працює

2. Підготовка середовища та перший .vbs

3. Синтаксис, коментарі, рядки та продовження рядка

4. Змінні та типи Variant

5. Оператори та вирази

6. Умови If / Else / Select Case

7. Цикли For / For Each / Do

8. Масиви

9. Процедури, функції та область видимості

10. Обробка помилок

Частина II. Робота з об’єктами Windows

11. Об’єктна модель і COM

12. FileSystemObject: файли й папки

13. Текстові файли та кодування

14. WScript.Shell: запуск програм, змінні середовища, реєстр

15. WScript.Network: диски, принтери, мережеві ресурси

16. WMI: комп’ютер, процеси, служби та обладнання

17. WSH: аргументи командного рядка, StdIn/StdOut, .wsf

Частина III. Практична автоматизація

18. Логування та діагностика

19. Дата і час

20. Регулярні вирази RegExp

21. Робота з CSV та структурованими даними

22. HTTP/XML та COM-автоматизація

23. Excel/Word через COM

24. Поштові та мережеві задачі: межі VBScript

25. Безпечна автоматизація

Частина IV. Практикум

26. Мініпроєкт: резервне копіювання

27. Мініпроєкт: інвентаризація ПК

28. Мініпроєкт: контроль папки та архівування

29. Мініпроєкт: звіт у CSV

30. Мініпроєкт: запуск задачі з параметрами

31. Мініпроєкт: сервісний скрипт для адміністратора

32. Стиль, структура та повторне використання коду

33. Типові помилки та як їх шукати

34. VBScript → PowerShell: що переносити

35. Фінальний маршрут навчання

Додатки. Шпаргалки, словник, задачі та довідник класів

Вступ. Що ви отримаєте після проходження курсу

VBScript (Visual Basic Scripting Edition) — скриптова мова сімейства Visual Basic, історично тісно пов'язана з Windows Script Host (WSH), COM та автоматизацією Windows. У цьому самовчителі ми не вчимо "писати пару команд", а будуємо правильну модель мислення: дані → умови → цикли → процедури → об'єкти → помилки → логування → повноцінні інструменти.

Головний принцип: кожен розділ закінчується практичними прикладами. Не просто читайте код — запускайте його, змінюйте, навмисно ламайте й виправляйте.

> **Про актуальність:** Microsoft зараз рекомендує планувати міграцію з VBScript на PowerShell для сучасної автоматизації Windows. VBScript варто знати передусім там, де вже існує спадковий код, legacy-інфраструктура або потрібне розуміння старих WSH/COM-рішень.

1. Що таке VBScript і де він працює

VBScript — інтерпретована мова, а не окрема сучасна IDE-платформа. Типовий сценарій у Windows: створити файл з розширенням .vbs і запустити його через wscript.exe або cscript.exe. WScript орієнтований на сценарії з графічними повідомленнями, CScript — на консоль.

Компонент	Роль	Приклад
VBScript engine	Виконує код VBScript	vbscript.dll / системний рушій
WSH	Середовище запуску сценаріїв	wscript.exe, cscript.exe
COM	Доступ до об'єктів Windows/Office	CreateObject(...)
WMI	Керування та запит до системної інформації	Win32_Process, Win32_OperatingSystem
ADODB.Stream	Потокове читання/запис	робота з текстом/байтами
FileSystemObject	Файлова система	файли, папки, диски

Для навчання зручно використовувати cscript у командному рядку:

```
cscript //nologo hello.vbs
```

Запам'ятайте: VBScript ≠ VBA ≠ VB.NET. Синтаксис споріднений, але середовища та доступні об'єкти різні.

2. Підготовка середовища та перший .vbs

Створіть папку, наприклад C:\VBScriptCourse. Створіть файл hello.vbs у Блокноті. Для першого запуску використайте консоль, щоб бачити StdOut.

```
Option Explicit

WScript.Echo "Привіт, VBScript!"
```

1. Збережіть файл як hello.vbs, а не hello.vbs.txt.
2. Відкрийте cmd.exe.
3. Перейдіть у папку з файлом: cd /d C:\VBScriptCourse
4. Запустіть: cscript //nologo hello.vbs
5. Змініть текст і запустіть ще раз.

Краща звичка: Починайте кожен нетривіальний скрипт з Option Explicit. Це змусить оголошувати змінні і ловити частину помилок раніше.

3. Синтаксис, коментарі, рядки та продовження рядка

Коментар починається з апострофа. Одна інструкція зазвичай займає один рядок. Для продовження інструкції використовується пробіл + символ підкреслення.

```
' Коментар
Option Explicit
Dim name
name = "Valik"
WScript.Echo "Привіт, " & name
```

Рядки в VBScript обмежуються подвійними лапками. Щоб вставити лапки у рядок, їх дублюють.

```
WScript.Echo "Він сказав: ""Привіт!"""
```

Конкатенація рядків виконується оператором &. Не плутайте його з математичним +: явне використання & робить намір зрозумілішим.

4. Змінні та типи Variant

VBScript використовує динамічну типізацію. Змінна зазвичай має тип Variant і може містити число, текст, дату, Boolean, Null, Empty, масив або посилання на об’єкт. Це зручно, але може породжувати приховані перетворення.

```
Option Explicit
Dim a, b, total
a = 10
b = 25
total = a + b
WScript.Echo total
```

Концепція	Функції/значення	Коментар
Порожня змінна	Empty	Початковий стан неініціалізованого Variant
Відсутнє значення	Null	Спеціальне “немає значення”
Логічне	True / False	Boolean
Рядок	CStr	Перетворення у текст
Число	CLng, CDbl, CInt	Явне числове перетворення
Дата	CDate, DateSerial	Робота з датами
Тип	VarType / TypeName	Перевірка типу

```
Dim x
x = "123"
WScript.Echo IsNumeric(x)
WScript.Echo CLng(x)
```

5. Оператори та вирази

Основні математичні оператори: +, -, *, /, \, Mod, ^. Порівняння: =, <>, <, >, <=, >=. Логічні оператори: And, Or, Not, а також Eqv, Imp.

```
x = 10
y = 3
WScript.Echo x / y
WScript.Echo x \ y
WScript.Echo x Mod y
WScript.Echo x ^ 2
```

Пастка: Оператор = використовується і для присвоєння, і для порівняння — контекст визначає значення. Окремого ==, як у C-подібних мовах, немає.

6. Умови If / Else / Select Case

```
If age >= 18 Then
    WScript.Echo "Повнолітній"
Else
    WScript.Echo "Неповнолітній"
End If
Select Case status
    Case "OK"
        WScript.Echo "Готово"
    Case "WARN"
        WScript.Echo "Попередження"
    Case Else
        WScript.Echo "Невідомо"
End Select
```

Вкладайте умови так, щоб читач міг швидко побачити головний сценарій. Для великих дерев умов краще розділяти логіку на функції.

7. Цикли For / For Each / Do

```
Dim i
For i = 1 To 5
    WScript.Echo i
Next
Dim item
For Each item In Array("A", "B", "C")
    WScript.Echo item
```

```
Next
Dim n
n = 0
Do While n < 5
    WScript.Echo n
    n = n + 1
Loop
Do Until ready
    WScript.Sleep 1000
Loop
```

Безпека циклу: Завжди перевіряйте, що стан, який завершує цикл, реально змінюється. Інакше отримаєте нескінченний цикл.

8. Масиви

```
Dim nums
nums = Array(10, 20, 30)
WScript.Echo UBound(nums)
WScript.Echo nums(0)
Dim a(4)
a(0) = "one"
a(4) = "five"
```

Для динамічних масивів використовується ReDim, а збереження даних при зміні розміру — ReDim Preserve. Функції LBound/UBound допомагають безпечно обходити межі.

```
Dim data
ReDim data(2)
data(0) = "A"
data(1) = "B"
data(2) = "C"
ReDim Preserve data(4)
```

9. Процедури, функції та область видимості

```
Sub SayHello(name)
    WScript.Echo "Привіт, " & name
End Sub

Function Add(a, b)
    Add = a + b
```

```
End Function


Call SayHello("Anna")
WScript.Echo Add(2, 3)
```

Використовуйте Sub для дії, Function — для обчислення/повернення результату. Параметри передаються ByRef за замовчуванням; для явного значення можна вказати ByVal.

```
Sub Increment(ByRef x)
    x = x + 1
End Sub


Sub PrintCopy(ByVal x)
    WScript.Echo x
End Sub
```

10. Обробка помилок

VBScript має On Error Resume Next, On Error GoTo 0 і On Error GoTo -1 не має аналога як у VB.NET; типова модель — тимчасово дозволити очікувану помилку, перевірити Err, а потім повернути нормальну обробку.

```
On Error Resume Next
Set fso = CreateObject("Scripting.FileSystemObject")
Set f = fso.GetFile("C:\not-exists.txt")
If Err.Number <> 0 Then
    WScript.Echo "Помилка: " & Err.Number & " / " & Err.Description
    Err.Clear
End If
On Error GoTo 0
```

Правило: Не залишайте On Error Resume Next увімкненим на весь файл. Це приховує справжні помилки.

11. Об’єктна модель і COM

Сильна сторона VBScript у Windows — COM automation. Ви створюєте об’єкт через CreateObject, після чого звертаєтесь до його методів і властивостей.

```
Set shell = CreateObject("WScript.Shell")
shell.Echo "Діалог не є методом Shell; краще використати WScript.Echo"
Set fso = CreateObject("Scripting.FileSystemObject")
WScript.Echo fso.GetAbsolutePathName(".")
```

Термін	Що означає
Object	Посилання на об’єкт
Property	Властивість: стан/дані об’єкта

Термін	Що означає
Method	Метод: дія об’єкта
Set	Присвоєння об’єктного посилання
Nothing	Відсутність об’єктного посилання
CreateObject	Створення COM-об’єкта за ProgID

12. FileSystemObject: файли й папки

Scripting.FileSystemObject (FSO) — один із головних інструментів VBScript. Через нього можна створювати папки, перевіряти існування, копіювати, переміщати, видаляти файли та перебирати каталоги.

```
Option Explicit
Dim fso, folder
Set fso = CreateObject("Scripting.FileSystemObject")
If Not fso.FolderExists("C:\Temp") Then fso.CreateFolder "C:\Temp"
Set folder = fso.GetFolder("C:\Temp")
WScript.Echo folder.Path
If fso.FileExists("C:\Temp\test.txt") Then
    fso.CopyFile "C:\Temp\test.txt", "C:\Temp\backup.txt", True
End If
Dim file
For Each file In folder.Files
    WScript.Echo file.Name & " | " & file.Size
Next
```

13. Текстові файли та кодування

FSO.OpenTextFile зручний для простих ANSI/Unicode сценаріїв, але з сучасним UTF-8 та різними кодуваннями часто краще працювати через ADODB.Stream. Це важливо, якщо скрипт обробляє український текст.

```
Const ForReading = 1
Set ts = fso.OpenTextFile("C:\Temp\data.txt", ForReading, False)
text = ts.ReadAll
ts.Close
WScript.Echo text
Const ForWriting = 2
Set ts = fso.OpenTextFile("C:\Temp\log.txt", ForWriting, True)
ts.WriteLine "Start"
ts.Close
```

Для контрольованого UTF-8 корисний ADODB.Stream. Типовий шаблон: Type=2 (text), Charset="utf-8", Open, LoadFromFile/Write, SaveToFile.

```
Set stream = CreateObject("ADODB.Stream")
stream.Type = 2
stream.Charset = "utf-8"
stream.Open
stream.WriteText "Привіт, UTF-8!"
stream.SaveToFile "C:\Temp\utf8.txt", 2
stream.Close
```

14. WScript.Shell: запуск програм, змінні середовища, реєстр

WScript.Shell дає доступ до запуску процесів, змінних середовища, реєстру, спеціальних папок та клавіш/Shortcut-об’єктів.

```
Set shell = CreateObject("WScript.Shell")
shell.Run "notepad.exe", 1, True
userName = shell.ExpandEnvironmentStrings("%USERNAME%")
WScript.Echo userName
shell.RegWrite "HKCU\Software\VBScriptCourse\Demo", "Hello", "REG_SZ"
WScript.Echo shell.RegRead("HKCU\Software\VBScriptCourse\Demo")
shell.RegDelete "HKCU\Software\VBScriptCourse\Demo"
```

Обережність: Не пишіть у HKLM, системні служби чи критичні ключі реєстру без чіткого розуміння наслідків та прав доступу.

15. WScript.Network: диски, принтери, мережеві ресурси

```
Set net = CreateObject("WScript.Network")
WScript.Echo net.UserName
WScript.Echo net.ComputerName
WScript.Echo net.UserDomain
net.MapNetworkDrive "Z:", "\\server\share"
net.RemoveNetworkDrive "Z:", True, True
```

Для скриптів адміністрування важливо розрізняти локальний шлях, UNC-шлях і змонтований диск. Не припускайте, що процес, запущений як інший користувач або служба, бачить той самий набір мережевих дисків.

16. WMI: комп’ютер, процеси, служби та обладнання

WMI (Windows Management Instrumentation) дозволяє запитувати системну інформацію. У VBScript типовий шаблон: GetObject("winmgmts:...") → ExecQuery → For Each.

```
Set svc = GetObject("winmgmts:\\.\root\cimv2")
Set items = svc.ExecQuery("SELECT Caption, Version FROM Win32_OperatingSystem")
For Each os In items
    WScript.Echo os.Caption & " | " & os.Version
Next
Set items = svc.ExecQuery("SELECT Name, ProcessId FROM Win32_Process")
For Each proc In items
    WScript.Echo proc.Name & " | PID=" & proc.ProcessId
Next
Set services = svc.ExecQuery("SELECT Name, State, StartMode FROM Win32_Service")
For Each s In services
    If s.State = "Running" Then WScript.Echo s.Name
Next
```

Продуктивність: Не робіть SELECT * без причини. Витягуйте лише потрібні властивості та фільтруйте WHERE, коли це доречно.

17. WSH: аргументи командного рядка, StdIn/StdOut, .wsf

WScript.Arguments дозволяє будувати справжні командні інструменти.

```
Dim args
Set args = WScript.Arguments
If args.Count = 0 Then
    WScript.Echo "Використання: cscript tool.vbs <path>"
    WScript.Quit 2
End If
WScript.Echo "Шлях: " & args(0)
For i = 0 To WScript.Arguments.Count - 1
    WScript.Echo i & ": " & WScript.Arguments(i)
Next
```

WSH також має Named arguments: /name:value. Для складніших сценаріїв корисний Windows Script File .wsf, де можна мати кілька job та описувати host.

18. Логування та діагностика

Надійний скрипт має повідомляти не тільки про успіх, а й про те, де він працює, що саме робить і з яким результатом завершився. Зробіть маленьку функцію Log.

```
Sub Log(msg)
    Dim line, ts
    line = Now & " | " & msg
    Set ts = fso.OpenTextFile(logPath, 8, True)
    ts.WriteLine line
    ts.Close
End Sub
```

Для командних скриптів використовуйте cscript //nologo, повертайте коди завершення через WScript.Quit і відокремлюйте “людський” вивід від машинного, якщо результат буде споживатися іншим інструментом.

19. Дата і час

```
WScript.Echo Now
WScript.Echo Date
WScript.Echo Time
WScript.Echo Year(Now)
WScript.Echo Month(Now)
WScript.Echo Day(Now)
d = DateSerial(2026, 8, 29)
WScript.Echo DateAdd("d", 7, d)
WScript.Echo DateDiff("d", d, Date)
```

Порада: Не зберігайте дати в локалізованому форматі, якщо файл повинен читатися іншими системами. Формат YYYY-MM-DD HH:NN:SS значно безпечніший для журналів.

20. Регулярні вирази RegExp

VBScript має об’єкт RegExp для пошуку та заміни за шаблонами. Це особливо корисно для витягування ID, номерів, шляхів або перевірки простих форматів.

```
Set re = New RegExp
re.Pattern = "^[0-9]+$"
re.Global = False
WScript.Echo re.Test("12345")
Set re = New RegExp
re.Pattern = "[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}"
re.Global = True
Set matches = re.Execute(text)
For Each m In matches
    WScript.Echo m.Value
Next
```

Межі: RegExp у VBScript не є сучасним PCRE-двигуном. Для складних regex-конструкцій перевіряйте сумісність із конкретною реалізацією VBScript.

21. CSV та структуровані дані

CSV легко читати “простим Split”, тільки якщо поля гарантовано не містять ком. У реальному CSV можливі лапки, коми всередині полів та перенос рядка. Для простих внутрішніх форматів Split достатній; для складних — краще використовувати відповідний сучасний інструмент.

```
parts = Split(line, ",")
For i = 0 To UBound(parts)
    WScript.Echo Trim(parts(i))
Next
```

Для власних логів часто краще створити стабільний табличний формат: timestamp, level, action, path, result. Це спрощує подальший аналіз.

22. HTTP/XML та COM-автоматизація

VBScript може працювати з COM-класами, що надають HTTP або XML API, але конкретні ProgID та політики безпеки залежать від Windows і встановлених компонентів. Застарілі рішення можуть використовувати XMLHTTP/ServerXMLHTTP, ADODB.Stream та MSXML DOM.

```
Set http = CreateObject("MSXML2.XMLHTTP")
http.Open "GET", "https://example.com/", False
http.Send
WScript.Echo http.Status
WScript.Echo Left(http.responseText, 200)
```

Безпека: Не вимикайте перевірки TLS/сертифікатів “щоб працювало”. Для сучасних API краще розглядати PowerShell або іншу актуальну мову, якщо VBScript не є обов’язковим.

23. Excel/Word через COM

Одна з історично популярних причин існування VBScript — керування Office через COM. Загальна модель: CreateObject("Excel.Application") → Workbooks → Worksheets → Range.

```
Set excel = CreateObject("Excel.Application")
excel.Visible = False
Set book = excel.Workbooks.Add
Set sheet = book.Worksheets(1)
sheet.Cells(1,1).Value = "Hello"
book.SaveAs "C:\Temp\report.xlsx"
book.Close False
excel.Quit
Set sheet = Nothing
```

```
Set book = Nothing
Set excel = Nothing
```

Дуже важливо: Завжди закривайте Office COM-об’єкти. Інакше можна отримати завислі EXCEL.EXE / WINWORD.EXE та блокування файлів.

24. Поштові та мережеві задачі: межі VBScript

У старих середовищах VBScript часто використовували CDO, WMI, XMLHTTP та COM-провайдери для мережевої автоматизації. Але сучасні TLS, OAuth, Graph API, MFA та хмарні сервіси роблять такі рішення незручними. Для нового коду віддавайте перевагу PowerShell/API-клієнтам.

25. Безпечна автоматизація

Автоматизація має бути передбачуваною. Найнебезпечніші помилки — неправильний шлях, відсутність перевірки вводу, тихе ігнорування помилок, видалення даних без dry-run та запуск від надмірних привілеїв.

Практика	Що робити
Dry-run	Спочатку тільки показувати заплановані дії
Allowlist	Дозволяти лише очікувані каталоги/розширення
Backup	Перед змінами створювати копії
Logging	Записувати timestamp + action + result
Least privilege	Працювати з мінімальними правами
Validation	Перевіряти аргументи до виконання
Exit codes	Повернути 0 для успіху, ненульове для помилки

26. Мініпроєкт: резервне копіювання

Завдання: скрипт приймає два аргументи — source та destination; перевіряє існування, створює каталог призначення, копіює файли і записує журнал.

```
Option Explicit
Dim fso, args, src, dst
Set fso = CreateObject("Scripting.FileSystemObject")
Set args = WScript.Arguments
If args.Count < 2 Then
    WScript.Echo "Usage: cscript backup.vbs <source> <destination>"
    WScript.Quit 2
End If
src = args(0)
dst = args(1)
If Not fso.FolderExists(src) Then
    WScript.Echo "Source not found: " & src
    WScript.Quit 3
```

```
End If
If Not fso.FolderExists(dst) Then fso.CreateFolder dst
fso.CopyFolder src, dst, True
WScript.Echo "Backup complete."
```

Розширення для самостійної роботи: маскувати тимчасові файли, створювати підпапку з датою, перевіряти вільне місце, логувати кількість файлів.

27. Мініпроєкт: інвентаризація ПК

```
Option Explicit
Dim wmi, items, it
Set wmi = GetObject("winmgmts:\\.\root\cimv2")
Set items = wmi.ExecQuery("SELECT Caption, Version, OSArchitecture FROM Win32_OperatingSystem")
For Each it In items
    WScript.Echo "OS: " & it.Caption
    WScript.Echo "Version: " & it.Version
    WScript.Echo "Arch: " & it.OSArchitecture
Next
```

Додайте CPU через Win32_Processor, RAM через Win32_ComputerSystem, диски через Win32_LogicalDisk і мережеві адаптери через Win32_NetworkAdapterConfiguration.

28. Мініпроєкт: контроль папки та архівування

Алгоритм: знайти файли старші за N днів → перемістити у Archive\YYYY-MM → записати журнал. Ключова функція — DateDiff("d", file.DateLastModified, Now).

```
cutoff = DateAdd("d", -30, Now)
For Each file In folder.Files
    If file.DateLastModified < cutoff Then
        WScript.Echo "Архівувати: " & file.Path
    End If
Next
```

Порада: Спочатку зробіть режим “тільки показати”, потім додайте MoveFile. Так ви перевірите логіку без ризику втрати даних.

29. Мініпроєкт: звіт у CSV

Використовуйте WriteLine з FSO та правильним екрануванням CSV-полів. Мінімальна функція CsvEscape повинна укладати поле в подвійні лапки, якщо в ньому є кома, лапка або перенос рядка, і подвоювати внутрішні лапки.

```
Function CsvEscape(value)
```

```
    value = CStr(value)
    value = Replace(value, """", """""" )
    CsvEscape = """" & value & """"
End Function
```

30. Мініпроєкт: запуск задачі з параметрами

Скрипт може діяти як тонка оболонка: перевірити аргументи, сформувати команду і передати її іншій програмі. Для запуску використовуйте WScript.Shell.Run або Exec. Exec корисний, коли потрібно прочитати StdOut/StdErr.

```
Set sh = CreateObject("WScript.Shell")
Set ex = sh.Exec("cmd /c whoami")
Do While Not ex.StdOut.AtEndOfStream
    WScript.Echo ex.StdOut.ReadLine
Loop
```

31. Мініпроєкт: сервісний скрипт для адміністратора

Об’єднайте практики: аргументи → перевірки → WMI → лог → код завершення. Приклад сценарію: отримати список служб, що мають статус Stopped, але StartMode="Auto", та записати їх у CSV.

```
Set services = svc.ExecQuery("SELECT Name, State, StartMode FROM Win32_Service WHERE
StartMode='Auto'")
For Each s In services
    If s.State <> "Running" Then
        WScript.Echo s.Name & " | " & s.State
    End If
Next
```

32. Стиль, структура та повторне використання коду

Середній VBScript швидко перетворюється на “лапшу”, якщо всі дії написані зверху вниз. Хороша структура: конфігурація → вхідні аргументи → основний запуск → функції → обробка помилок → завершення.

```
Option Explicit

Const APP_NAME = "PcInventory"
Dim fso, shell

Set fso = CreateObject("Scripting.FileSystemObject")
Set shell = CreateObject("WScript.Shell")
```

```
Main


Sub Main()
    ' Основний сценарій
End Sub
```

Уникайте глобальних змінних без потреби. Імена процедур робіть дієслівними: GetComputerInfo, WriteLog, CopyFiles, ValidateArgs.

33. Типові помилки та як їх шукати

Симптом	Ймовірна причина	Що перевірити
Object required	Об’єкт не створений або Set пропущено	Set obj = CreateObject(...)
Type mismatch	Невдале перетворення типу	VarType, IsNumeric, CStr/CLng
Permission denied	Немає прав	шлях, UAC, обліковий запис
File not found	Неправильний шлях/поточна папка	GetAbsolutePathName, Exists
Скрипт “мовчить”	Помилка прихована Resume Next	Err.Number/Description
Office не закрився	COM-об’єкти не звільнені	Close/Quit/Set Nothing
Нескінченний цикл	Умова не змінюється	лічильник та стан завершення

Техніка дебагу: звузити проблему до мінімального прикладу; додати WScript.Echo у ключові точки; перевірити Err одразу після ризикованого виклику; записати фактичні шляхи/значення у лог.

34. VBScript → PowerShell: що переносити

VBScript корисно знати як базу для міграції. Логіка часто переноситься напряму: змінні, умови, цикли, функції, файли та WMI-концепції. Але PowerShell має об’єкти та pipeline як центральну модель, значно кращу підтримку сучасних API, модулів і Windows-адміністрування.

VBScript	PowerShell (концептуально)
WScript.Echo	Write-Output / Write-Host
CreateObject	New-Object / .NET / cmdlets
FSO	Get-ChildItem, Copy-Item, Remove-Item, Get-Content
WMI ExecQuery	Get-CimInstance
WScript.Shell.Run	Start-Process
Environment variables	$env:NAME
Err	try/catch + $Error
Array/Dictionary patterns	масиви, hashtable, PSCustomObject

Стратегія: Не переписуйте механічно рядок у рядок. Спочатку визначте, яку задачу виконує скрипт, потім реалізуйте її сучасними cmdlet/класами PowerShell.

35. Фінальний маршрут навчання

Після прочитання цього самовчителя зробіть 10 повторень: кожне завдання напишіть з нуля без копіювання, потім порівняйте з прикладом. Мета — навчитися бачити задачу як набір маленьких функцій.

6. Написати скрипт, який рахує файли у папці.
7. Додати фільтр за розширенням.
8. Вивести сумарний розмір.
9. Додати журнал.
10. Додати аргументи командного рядка.
11. Додати обробку помилок.
12. Додати CSV-звіт.
13. Додати WMI-інформацію про ПК.
14. Додати dry-run.
15. Переписати готовий інструмент на PowerShell і порівняти код.

36. Повний довідник вбудованих функцій

Навіть якщо ви не будете пам’ятати всі функції, корисно знати їхні групи. Це дозволяє швидко знаходити коротке рішення замість написання власного коду.

Група	Функції	Призначення
Рядки	Len, Left, Right, Mid, InStr, InStrRev	Довжина, вирізання та пошук
Рядки	Replace, Trim, LTrim, RTrim, UCase, LCase	Заміна та нормалізація
Рядки	Split, Join, String, Space, StrReverse	Поділ/об’єднання та генерація
Типи	VarType, TypeName, IsNumeric, IsDate, IsEmpty, IsNull, IsObject	Перевірка значень
Числа	Abs, Sgn, Int, Fix, Round, Rnd, Randomize	Математичні операції
Перетворення	CStr, CInt, CLng, CDbl, CSng, CDate, CBool	Явні перетворення
Дата/час	Now, Date, Time, Year, Month, Day, Hour, Minute, Second	Компоненти дати й часу
Дата/час	DateAdd, DateDiff, DatePart, DateSerial, TimeSerial	Обчислення та створення дат
Масиви	Array, LBound, UBound, IsArray, Filter	Робота з масивами
Файл/система	CurDir, ChDrive, ChDir, GetObject	Поточний каталог та об’єкти
Службові	Err, RGB, Asc, Chr, ChrW, AscW	Помилки, символи та значення

Порівнюйте функцію з задачею. Наприклад, якщо треба видалити пробіли по краях — Trim, а не Replace з десятком шаблонів.

37. Рядки: детальна практика

```
s = "  Hello, World!  "
WScript.Echo Len(s)
WScript.Echo Trim(s)
WScript.Echo UCase(s)
WScript.Echo Left(s, 5)
WScript.Echo Right(s, 6)
WScript.Echo Mid(s, 4, 5)
WScript.Echo InStr(1, s, "World", vbTextCompare)
```

У VBScript немає повноцінного типу Char; один символ — це теж рядок довжини 1. Asc/Chr працюють з кодами символів, а AscW/ChrW — з Unicode-кодами.

```
ch = Chr(65)
WScript.Echo ch
WScript.Echo Asc("A")
WScript.Echo ChrW(1041)
```

38. Числа, округлення та арифметичні пастки

```
WScript.Echo 10 / 4
WScript.Echo 10 \ 4
WScript.Echo 10 Mod 4
WScript.Echo Round(12.3456, 2)
WScript.Echo Int(-1.2)
WScript.Echo Fix(-1.2)
```

Int округлює вниз до меншого цілого, а Fix відкидає дробову частину до нуля. Для від’ємних чисел це дає різні результати.

Не покладайтеся на неявні перетворення. Перед арифметикою перевіряйте IsNumeric і за потреби використовуйте CLng/CDbl.

39. Dictionary: ключ → значення

Scripting.Dictionary — дуже корисний COM-об’єкт для таблиць відповідностей, підрахунків і швидкого пошуку за ключем.

```
Set dict = CreateObject("Scripting.Dictionary")
dict.Add "UA", "Україна"
dict.Add "GB", "United Kingdom"
WScript.Echo dict("UA")
WScript.Echo dict.Exists("US")
For Each key In dict.Keys
```

```
    WScript.Echo key & " -> " & dict(key)
Next
dict("count") = 1
dict("count") = dict("count") + 1
```

Практика: Dictionary зручний для лічильників: кожен знайдений елемент збільшує значення за його ключем.

40. Класи та власні об’єкти

VBScript підтримує Class...End Class. Це не об’єктна система сучасного VB.NET, але її достатньо для простих моделей даних і сервісних обгорток.

```
Class Person
    Public Name
    Public Age

    Public Sub SayHello()
        WScript.Echo "Мене звати " & Name
    End Sub
End Class

Dim p
Set p = New Person
p.Name = "Anna"
p.Age = 30
p.SayHello
```

Клас особливо корисний, коли функціям потрібно передавати не десяток розрізнених параметрів, а один об’єкт із логічно пов’язаними даними.

41. Колекції, об’єкти та властивості за замовчуванням

Багато COM-об’єктів повертають колекції. Їх типова форма: For Each item In collection. Не припускайте, що всі колекції мають однакові методи.

```
For Each file In folder.Files
    WScript.Echo file.Name
Next
```

У документації до конкретного COM-класу шукайте: Properties, Methods, Collections, Return type та Required parameters. Це швидше, ніж експериментувати із випадковими викликами.

42. GetObject, прив’язка до існуючих об’єктів і файлів

CreateObject створює новий COM-об’єкт. GetObject може отримати вже наявний об’єкт або підключення до системи, наприклад WMI.

```
Set wmi = GetObject("winmgmts:\\.\root\cimv2")
```

Для Office також зустрічається патерн GetObject, коли треба під’єднатися до уже запущеного екземпляра. Такий підхід треба використовувати свідомо: стан уже відкритого об’єкта може бути непередбачуваним.

43. WScript.Shell: Run проти Exec

Метод	Коли використовувати	Особливість
Run	Потрібно лише запустити команду	Може чекати завершення, але не дає зручного StdOut/StdErr
Exec	Потрібно читати StdOut/StdErr	Повертає WshScriptExec
AppActivate	Потрібно активувати вікно	Залежить від назви/вікна, крихке рішення
SendKeys	Потрібно імітувати клавіатуру	Крихке; непридатно як надійний UI API

```
Set sh = CreateObject("WScript.Shell")
Set p = sh.Exec("cmd /c ipconfig /all")
output = p.StdOut.ReadAll
WScript.Echo output
```

Правило: Для автоматизації GUI краще шукати API/COM/CLI програми, а не керувати вікнами через SendKeys.

44. Змінні середовища та спеціальні папки

```
Set sh = CreateObject("WScript.Shell")
Set env = sh.Environment("PROCESS")
WScript.Echo env("PATH")
WScript.Echo sh.ExpandEnvironmentStrings("%TEMP%")
```

WshShell.SpecialFolders може повертати спеціальні системні папки, але конкретний набір залежить від WSH/Windows. Перевіряйте доступність ключа перед використанням.

45. Реєстр: типи значень і безпечне читання

Тип	Значення RegWrite
REG_SZ	Рядок
REG_EXPAND_SZ	Рядок з %ENV%
REG_DWORD	32-бітне число
REG_BINARY	Масив байтів

```
On Error Resume Next
```

```
value = shell.RegRead("HKCU\Software\VBScriptCourse\Missing")
If Err.Number <> 0 Then
    WScript.Echo "Ключ/значення відсутнє"
    Err.Clear
End If
On Error GoTo 0
```

Для читання робіть окрему безпечну функцію RegistryRead, яка повертає IsSuccess і значення. Це краще, ніж розсипати Resume Next по всьому коду.

46. WMI: глибше — фільтрація, властивості та методи

```
query = "SELECT Name, ProcessId, ExecutablePath FROM Win32_Process WHERE Name = notepad.exe"
Set items = svc.ExecQuery(query)
For Each proc In items
    WScript.Echo proc.Name
    WScript.Echo proc.ProcessId
Next
```

WMI-клас складається з властивостей і методів. Наприклад, деякі класи процесів дозволяють викликати Terminate. Не використовуйте дії зміни стану, доки повністю не перевірите цільовий об’єкт.

```
Set diskSet = svc.ExecQuery("SELECT DeviceID, FreeSpace, Size FROM Win32_LogicalDisk WHERE
DriveType=3")
For Each d In diskSet
    WScript.Echo d.DeviceID & " | free=" & d.FreeSpace & " | size=" & d.Size
Next
```

47. WMI віддалено

Історично WMI дозволяє підключатися до іншого комп’ютера. У реальному середовищі це залежить від прав, firewall, DCOM/WMI policy та облікових даних. Не будуйте інструкції, які обходять політики доступу.

```
Set svc = GetObject("winmgmts:{impersonationLevel=impersonate}\remote-pc\root\cimv2")
```

Адміністрування: Для нових систем керування віддаленими хостами Microsoft рекомендує сучасні PowerShell/CIM-підходи, а не створення нової WMI-архітектури на VBScript.

48. Active Directory та ADSI: що треба знати

VBScript у корпоративних Windows-середовищах часто використовував ADSI/LDAP для роботи з Active Directory. Конкретні можливості залежать від домену та прав. Типові ProgID включають ADsDSOObject для запитів та IADs-об’єкти для роботи з каталогом.

```
Set rootDSE = GetObject("LDAP://RootDSE")
WScript.Echo rootDSE.Get("defaultNamingContext")
```

Цей приклад лише показує модель доступу. Запити до каталогу слід писати з урахуванням продуктивності, paging, фільтрів LDAP та принципу мінімальних прав.

49. Планувальник завдань та автоматичний запуск

VBScript сам по собі не є сучасним планувальником. Найпростіше legacy-рішення — створити скрипт і запускати його через Task Scheduler або викликати schtasks.exe. Скрипт має бути idempotent: повторний запуск не повинен ламати результат.

```
Set sh = CreateObject("WScript.Shell")
cmd = "schtasks /Run /TN "MyTask""
sh.Run cmd, 0, True
```

Звертайте особливу увагу на обліковий запис, права та робочу директорію задачі. Те, що працює подвійним кліком, може не працювати з Task Scheduler.

50. Робоча директорія, абсолютні та відносні шляхи

```
Set fso = CreateObject("Scripting.FileSystemObject")
WScript.Echo "Current: " & fso.GetAbsolutePathName(".")
WScript.Echo "Script: " & WScript.ScriptFullName
```

Надійний скрипт майже завжди перетворює вхідний шлях в абсолютний і не покладається на поточну робочу папку. Особливо це важливо для Task Scheduler, служб та запуску з інших програм.

51. Конфігурація без захардкожених значень

Не зашивайте десятки шляхів, порогів і режимів у код. Навіть простий файл key=value може суттєво полегшити підтримку.

```
Function ReadConfig(path)
    Dim cfg, line, parts
    Set cfg = CreateObject("Scripting.Dictionary")
    Set ts = fso.OpenTextFile(path, 1, False)
    Do Until ts.AtEndOfStream
        line = Trim(ts.ReadLine)
        If Len(line) > 0 Then
            If Left(line, 1) <> "#" Then
                parts = Split(line, "=", 2)
                If UBound(parts) = 1 Then cfg(Trim(parts(0))) = Trim(parts(1))
            End If
        End If
    Loop
    ts.Close
```

```
    Set ReadConfig = cfg
End Function
```

52. Ідемпотентність

Ідемпотентний скрипт можна запускати багато разів без накопичення пошкоджень. Наприклад, “створити папку, якщо її немає” — ідемпотентна операція. “Дописувати один і той самий рядок у файл при кожному запуску” — потенційно неілемпотентна.

Погано	Краще
Створювати папку без перевірки	FolderExists → CreateFolder
Копіювати без контролю результату	FileExists + CopyFile + лог
Безумовно додавати конфіг	Перевіряти, чи запис уже існує
Вбивати процес за назвою	Перевіряти PID/власника/критерії

53. Тестування VBScript

Навіть маленькі скрипти можна тестувати. Розділіть логіку на чисті функції, а операції з файловою системою винесіть в окремі процедури. Тестуйте окремо: валідний ввід, відсутній файл, порожній файл, кириличний текст, дуже великий файл, недостатні права.

16. Створіть тестову папку, а не використовуйте робочу.
17. Підготуйте вхідні файли різних типів.
18. Перевірте очікуваний результат.
19. Перевірте код завершення.
20. Перевірте лог.
21. Повторіть запуск і переконайтеся в ідемпотентності.

54. Продуктивність

VBScript нормальний для невеликих адміністративних задач, але слабший за сучасні інструменти на великих об’ємах. Найчастіші вузькі місця — багато COM-викликів, WMI-запити без фільтра, часті відкриття/закриття файлів та робота з великими текстами через конкатенацію.

```
Set ts = fso.OpenTextFile(logPath, 8, True)
For i = 1 To 1000
    ts.WriteLine i & " | record"
Next
ts.Close
```

Відкривати файл один раз на 1000 рядків значно логічніше, ніж відкривати/закривати його на кожній ітерації.

55. Пам’ять і звільнення COM-об’єктів

Використовуйте Set obj = Nothing для явного відпускання COM-посилань у довгих сценаріях. Особливо важливо для Office і великих колекцій.

```
Set excel = CreateObject("Excel.Application")
' ... work ...
book.Close False
excel.Quit
Set sheet = Nothing
Set book = Nothing
Set excel = Nothing
```

56. WSF: кілька скриптових компонентів в одному файлі

Windows Script File (.wsf) — XML-контейнер для одного або кількох script jobs. Він дозволяє вказувати мову, ім’я job і сценарії. Це корисно в legacy-рішеннях, де один запуск має виконувати пов’язаний набір компонентів.

```
<job id="Demo">
  <script language="VBScript">
    WScript.Echo "Hello from WSF"
  </script>
</job>
```

Примітка: Показаний фрагмент — концептуальний приклад структури. Для реального .wsf перевіряйте XML-синтаксис, екранування та конкретну версію WSH.

57. WSH Named Arguments та CLI-дизайн

Іменовані аргументи WScript.Arguments.Named дозволяють зробити інструмент читабельнішим: /source:C:\Data /dest:C:\Backup /dryrun.

```
Set named = WScript.Arguments.Named
If named.Exists("source") Then WScript.Echo named("source")
If named.Exists("dryrun") Then WScript.Echo "Dry-run requested"
```

Добрий CLI має -- або /? довідку, зрозумілі помилки, коди завершення та стабільні імена параметрів.

58. Коди завершення та інтеграція

```
If success Then
    WScript.Quit 0
Else
    WScript.Quit 1
End If
```

Код	Приклад значення	Сенс
0	Успіх	Операція виконана
1	Загальна помилка	Невизначена/непередбачена помилка
2	Невірні аргументи	Помилка використання
3	Не знайдено	Відсутній ресурс

Не існує єдиного універсального стандарту кодів для VBScript; головне — задокументувати значення та використовувати їх послідовно.

59. Документування legacy-коду

Коли ви зустрічаєте старий .vbs, перш ніж щось змінювати, запишіть: де запускається, від якого акаунта, які аргументи, які файли змінює, які COM-компоненти використовує, які коди завершення має та хто залежить від результату.

```
' Purpose: clean old logs
' Input: /path:<folder> /days:<N>
' Output: exit code 0/1
' WARNING: never deletes outside configured root
```

60. Підсумковий навчальний проєкт

Створіть один “реальний” інструмент з нуля: SysReport.vbs. Він приймає /out:path /csv /dryrun, збирає ОС, CPU, RAM, диски, запущені служби, кількість файлів у вказаній папці, записує лог і код завершення.

22. Спочатку спроєктуйте аргументи.
23. Потім створіть ValidateArgs.
24. Далі — GetOsInfo, GetCpuInfo, GetDiskInfo.
25. Додайте WriteCsv та WriteLog.
26. Додайте dry-run для небезпечних операцій.
27. Протестуйте на тестовій машині/віртуальній машині.
28. Зафіксуйте приклад очікуваного результату.
29. Порівняйте реалізацію з аналогом PowerShell.

Після цього ви вже не просто знаєте синтаксис VBScript — ви вмієте читати, підтримувати та будувати практичні WSH-скрипти.

Додаток A. Шпаргалка VBScript

Конструкція	Приклад
Оголошення	Dim x
Константа	Const ForReading = 1
Присвоєння	x = 10
Об’єкт	Set fso = CreateObject("Scripting.FileSystemObject")
Умова	If x > 0 Then ... End If
For	For i = 1 To 10 ... Next

Конструкція	Приклад
For Each	For Each f In folder.Files ... Next
Do	Do While cond ... Loop
Функція	Function Add(a,b) ... End Function
Підпрограма	Sub Log(msg) ... End Sub
Вихід	WScript.Quit 1
Очікування	WScript.Sleep 1000
Помилка	On Error Resume Next
Помилка назад	On Error GoTo 0
Текст	CStr(value)
Число	CLng(value), CDbl(value)
Дата	CDate(value), DateAdd, DateDiff
Масив	Array(1,2,3), UBound
Рядок	Len, Left, Right, Mid, InStr, Replace, Trim, Split, Join

Додаток B. Часто вживані COM-об’єкти

ProgID / механізм	Для чого
Scripting.FileSystemObject	Файли, папки, диски
WScript.Shell	Run, Exec, env, registry
WScript.Network	мережеві диски, комп’ютер/користувач
WScript.Dictionary	ключ-значення
VBScript.RegExp	регулярні вирази
ADODB.Stream	потоки, текст/байти
MSXML2.XMLHTTP	HTTP-запити в legacy-рішеннях
WMI / winmgmts	системні дані й адміністрування
Excel.Application	Excel automation
Word.Application	Word automation

Додаток C. Практичні задачі без розв’язку

1. Знайти всі .log файли, старші за 14 днів, і вивести їх розмір.

2. Порахувати кількість файлів у кожній підпапці заданого каталогу.

3. Створити структуру backup\YYYY-MM-DD і скопіювати туди вибрані файли.

4. Прочитати текстовий файл і порахувати кількість рядків, слів та символів.

5. Знайти в каталозі всі email-адреси регулярним виразом.

6. Отримати список запущених процесів через WMI і записати у CSV.

7. Отримати список автоматичних, але зупинених служб.

8. Створити скрипт, який перевіряє наявність вільного місця на диску.

9. Написати універсальну функцію Log(message, level).

10. Створити скрипт, який запускає зовнішню програму та очікує її завершення.

11. Зробити dry-run для операції переміщення файлів.

12. Створити звіт по Office-документах у папці.

13. Зробити простий конфігураційний файл key=value і функцію читання конфігурації.

14. Розробити CLI з /source: /dest: /dryrun.

15. Мігрувати один свій VBScript на PowerShell.

Додаток D. Глосарій

Термін	Пояснення
VBScript	Скриптова мова Microsoft сімейства Visual Basic.
WSH	Windows Script Host — середовище виконання скриптів.
COM	Component Object Model — модель компонентів/автоматизації Windows.
ProgID	Текстовий ідентифікатор COM-класу, наприклад Scripting.FileSystemObject.
WMI	Windows Management Instrumentation.
Variant	Основний динамічний тип значень VBScript.
ByRef	Параметр передається з можливістю змінити вихідну змінну.
ByVal	Передача значення без зміни вихідної змінної.
StdOut	Стандартний вихід процесу.
StdErr	Стандартний потік помилок.
UNC	Мережевий шлях виду \\server\share.
Legacy	Застаріле/успадковане рішення, яке може залишатися в експлуатації.
Dry-run	Режим, у якому скрипт показує дії, але не вносить зміни.

Додаток E. Офіційні та рекомендовані джерела

Оскільки VBScript є legacy-технологією, офіційна документація розкидана між історичними сторінками Microsoft Learn. Для довідки використовуйте насамперед Microsoft Learn та документацію WSH/WMI/COM. Сучасну автоматизацію Windows вивчайте паралельно з PowerShell.

Microsoft Learn: Deprecated features in Windows client — сторінка зі статусом VBScript та планом його майбутнього вилучення.

Microsoft Learn: Using VBScript — історична документація щодо VBScript та WSH.

Microsoft Learn: Visual Basic language reference — корисна як довідкове джерело для спорідненого синтаксису, але не сприймайте її як повну специфікацію VBScript.

Додаток F. Контрольний список готового скрипту

□ Є Option Explicit.

□ Аргументи перевіряються.

□ Немає зайвого On Error Resume Next.

□ Шляхи та імена файлів перевіряються.

□ Ризикові операції мають dry-run або резервну копію.

□ У логах є дата/час і результат.

□ Помилки завершують скрипт ненульовим кодом.

□ COM-об’єкти закриваються/звільняються.

□ Немає залежності від поточної робочої папки без потреби.

□ Код можна прочитати через місяць без пояснень автора.